Anomaly Detection Using Gaussian Mixture Probability Model to Implement Intrusion Detection System

Roberto Blanco[1,2], Pedro Malagón[1,2(✉)], Samira Briongos[1,2], and José M. Moya[1,2]

[1] LSI-Universidad Politecnica de Madrid, Madrid, Spain
[2] CCS-Center for Computational Simulation, Madrid, Spain
{r.bandres,malagon,samirabriongos,josem}@die.upm.es

Abstract. Network intrusion detection systems (NIDS) detect attacks or anomalous network traffic patterns in order to avoid cybersecurity issues. Anomaly detection algorithms are used to identify unusual behavior or outliers in the network traffic in order to generate alarms. Traditionally, Gaussian Mixture Models (GMMs) have been used for probabilistic-based anomaly detection NIDS. We propose to use multiple simple GMMs to model each individual feature, and an asymmetric voting scheme that aggregates the individual anomaly detectors to provide. We test our approach using the NSL dataset. We construct the normal behavior models using only the samples labelled as normal in this dataset and evaluate our proposal using the official NSL testing set. As a result, we obtain a F1-score over 0.9, outperforming other supervised and unsupervised proposals.

Keywords: Intrusion Detection · Gaussian Mixture Model · Voting

1 Introduction

In addition to general security concerns, service providers have to deal with attacks to their infrastructures, which can affect their service availability, their clients or industrial privacy, integrity or reliability of their solutions. Moreover, the irruption of the Internet of Things has lead to an exponential growth of the number of devices connected to the Internet. The challenges related to protect services, networks and devices are drastically increasing in complexity.

Rule-based protection mechanisms, such as firewalls, are not as effective as Intrusion Detection Systems (IDS) [5] when dealing with new security threats and complex systems. Intrusion Detection Systems are based on the assumption that an attack or an intrusion will change the pattern of resource usage or network flow. Traditionally, IDS are classified as signature-based or anomaly-based [1] depending on how they face detection. Signature-based detectors check if the collected samples match with known attacks, whereas anomaly-based detectors

H. Pérez García et al. (Eds.): HAIS 2019, LNAI 11734, pp. 648–659, 2019.
https://doi.org/10.1007/978-3-030-29859-3_55

build statistical models that characterize normal behavior and look for abnormal patterns. In practice, both approaches require to monitor network packets or to collect representative samples of the system they want to protect.

Training and evaluating a NIDS requires a comprehensive dataset that is representative of real traffic packets passing through a firewall. This dataset must contain normal and abnormal samples. Indeed, each sample of these datasets usually contains multiple features and its corresponding label. There are multiple datasets available for research purposes [15,17] which are commonly used to train IDS and test their performance, efficiency and accuracy.

The IDS classifies the data into categories using different methods. Multiple machine learning algorithms have been proposed for implementing the classifier of the IDS, including both supervised and unsupervised algorithms. Supervised algorithms are capable of detecting known varieties of attacks. However, new or undocumented attacks may go undetected. For this reason, it is commonly suggested to implement IDS based on anomaly detection algorithms. Besides, building a labelled dataset to model the new Internet of Things applications, including normal traffic and attacks, can be more expensive (or even impossible) than generating one with only normal patterns.

Our proposal is, consequently, based on anomaly detection [8]. We use Gaussian Mixture Models (GMM) [20] to model normal behavior. We propose a set of classifiers, which evaluate the individual probability of each of the features of a sample, to be considered normal according to GMM. This information is then used as the input to a voting based aggregation method which decides if the sample is normal or abnormal. This method does not require any anomalous sample during the training phase.

We use the NSL-KDD [17] dataset for our experiments. Our approach obtains an F1-score over 0.9 using a test set with completely new traces which are not related to the training set, neither normal nor attacks. We compare our solution to existing supervised and unsupervised methods, and the voting GMM outperforms most of the considered algorithms.

2 Related Work

The main goal of an anomaly detection algorithm is to filter out outliers. This task is critical in many disciplines, including medical diagnosis, fault prediction, fraud detection or network intrusion detection [10].

According to Domingues et al. [6] anomaly detection algorithms are divided into different families. Probabilistic methods fit the behavior of the system in a set of known functions (Gaussian with GMM [20] or other generic functions in Kernel Density Estimators, KDE [12]). Distance-based algorithms, such as the Local Outlier Factor (LOF) [4], are applied to Gaussian models or clusters of neighbors (using K-means or K-nearest neighbors). Neural networks constitute another family in which the most common type of network used for anomaly detection is known as Self-Organizing-Maps (SOM) [21]. Finally, domain-based algorithms, such as one-class Support Vector Machines (SVM) [22] have been used to establish an irregular multi-dimensional boundary to the normal data.

Network intrusion detection systems (NIDS) have been evaluated with multiple and well-known datasets. The KDD99 dataset [11] was used in [13] considering new attacks. The NSL-KDD dataset was evaluated in [19] and the UNSW-NB15 dataset in [16]. NSL-KDD includes a separate official testing set, with traces of attacks and normal traffic not present in the training set, which makes it ideal for testing anomaly detection algorithms.

GMM based algorithms have been proposed to implement NIDS in [2]. In [14] fuzzy logic was used to implement clustering using GMM. In [3], they present a method that uses a lower dimensional space and adapts to changes in time. A majority voting scheme is used in [9] with votes in time windows to reduce noise in outlier detection. These algorithms, in order to be applied in real environments, need to be fast and select realistic features. For example, Dromard et al. [7] use a clustering anomaly detection algorithm to meet real time constraints.

3 Materials and Methods

3.1 Dataset Description

NSL-KDD dataset [17] is a refined version of the KDD cup 99 dataset (a well known benchmark for the research of Intrusion Detection techniques). It contains the essential records of its predecessor balancing the proportion of normal versus attack traces, and excluding redundant records. Each record is composed of 41 attributes unfolding four different types of features of the flow, and its assigned label which classifies it as an attack or as normal. These features include basic characteristics of each network connection vector such as the duration or the number of bytes transferred, content related features like the number of "root" accesses, contextual time related traffic features such as the number of connections to the same destination, and host based traffic features like the number of connections to the same port number. The whole amount of records covers one normal class and four attack classes grouped as denial of service (DoS), surveillance (Probe), unauthorized access to local super user (R2L) and unauthorized access from a remote machine (U2R).

3.2 Data Preprocessing

NSL-KDD dataset contains numeric and categorical features. The most convenient method for managing categorical features when feeding them to machine learning algorithms is the one hot encoding conversion. However, in this dataset there are only three categorical features (protocol, service and flag) that are not independent from each other. We have removed the service and flag and we have only selected tcp traces for our experiments, as this is the most relevant and abundant protocol. Moreover, our intention is to build an anomaly detection model that could be applied in a router node of the network, so we have also removed the content related features which the router should not be able to reach. After this process, the number of data features has been reduced to 24

and we have a train dataset including 53600 normal and 49040 attack records and a test dataset with 7842 normal and 10971 attack traces. It is important to mention that the test dataset includes attacks that have not being included in any entry of the training set.

3.3 Normalization

Since the range of values of the raw data varies widely, normalization is a must step for some machine learning algorithms. It allows to calculate distances between points using the Euclidean distance or even accelerates the convergence of many optimization algorithms such as gradient descent. We use feature scaling to adjust all column feature values into the range [0,1] and avoid large variations in data.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

3.4 Principal Component Analysis

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of linearly uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component has, in turn, the highest possible variance under the constraint that it is orthogonal to the preceding components. This technique is mostly used to reduce the dimensionality of the data while preserving the maximum amount of information among the features. The problem to solve becomes much simpler, as it deals with less features and the solution is still good enough. In this work, we are mainly interested in the capability of PCA to obtain uncorrelated features. Our original space has not too many dimensions, for this reason we do not care about dimensionality reduction. We propose to use the PCA technique in order to make a transformation that allows to consider each generated feature as if it was independent from the others. We know it is not really true but it is a better approximation if we make the assumption after the PCA. We have explored both approaches.

3.5 Feature Gaussian Mixture Probability Model

A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters (Fig. 1 left). For a given set of data we can apply an expectation-maximization statistical iterative algorithm and obtain which points come from which Gaussian latent component. The algorithm provides a classification of the points and the latent components, which are useful to our approach. Our goal is to implement an anomaly detection algorithm by modeling the normal behavior of a system. Assuming that every feature of our

system normal traffic follows a Gaussian mixture distribution, we are able to obtain its latent components; i.e., we can estimate the mean and the variance of every Gaussian in the mixture. We consider an algorithm to distinguish normal from anomalous traffic using the obtained normal model. The simple Gaussian mixture model only gives us the probability of each sample to belong to every latent component of the mixture, but this is only useful when classifying and in our problem we don't know what an anomaly is and we should not use any attack record in the model building step. Therefore, we cannot use explicitly this model to detect any behavior different from the normal one. This is the reason why we need to obtain the latent components, because with them we can compute a probability of occurrence. With the assumption of the normal traffic in our system following a Gaussian mixture distribution we can obtain for a given traffic vector the latent component that each one of the vector features belongs with the highest probability. As we have characterized the latent components we can then compute the probability of occurrence for this traffic vector values following the corresponding latent components as if it was the worst case, that means that we compute the area under the normal curve for all the possible values with an absolute value greater than the analyzed value. For example, if we analyze the probability of occurrence for a value that match the mean of the latent component, this would be 1; on the other hand, if we consider the probability of occurrence for a value that match the mean plus the variance of the latent component, its probability wouldl be $1 - (0.3413 + 0.3413) = 0.3174$ (Fig. 1 right).

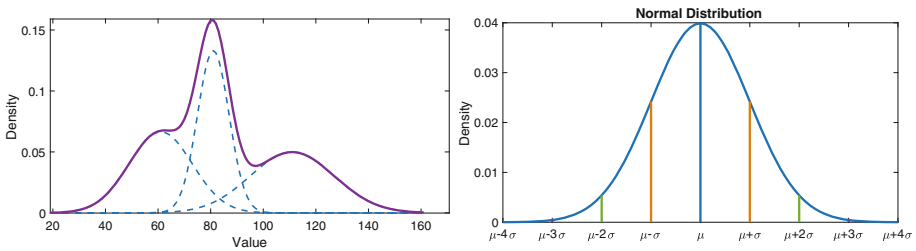

Fig. 1. Gaussian Mixture and Gaussian probabilities.

3.6 Probability Voting Scheme

The main point in this paper is the Feature Gaussian Mixture Probability Model that is supposed to obtain the occurrence probability of a certain value of a feature in a given traffic vector. This is in fact a probabilistic statistical model in which we expect that the normal values have a higher probability than the anomaly ones. In order to make decisions we need a method that aggregates all the features probabilities and applies a certain threshold so we can classify the entire traffic vector as normal or anomalous. Taking in account this idea we propose a simple voting scheme that evaluates each feature probability independently and then estimates the nature of the traffic vector based on the number

of independent positive evaluations. Our method needs two hyperparameters, one for establishing the individual feature probability threshold and the other as the minimum number of anomalous features to consider the whole vector as an attack. We have called the first one as α and it is the percentage error we can afford when classifying normal features. For every normal feature in our training dataset, we compute the occurrence probability. The value of alpha represents the percentage of the training normal feature probabilities that will be considered as anomalous for the model, so the decision threshold will be set as the maximum occurrence probability in the $1 - \alpha$ remaining percentage. For simplicity this percentages are normalized from 0 to 1. We have called the second one as consensus and it is just the number of positive (feature probability larger than the threshold) evaluations needed to consider the whole traffic vector as anomalous.

3.7 Other Machine Learning Algorithms

In order to compare the proposed methods, other state-of-art algorithms are introduced.

K-Means is an unsupervised learning algorithm that is mostly used for clustering. Given a set of data vectors with the same number of features (dimensionality d) and a number of desired clusters c, the algorithm is able to seek and find the optimum c points in the space d that minimize the sum of squared distances of the whole set of data vectors to its closest point. At the end this means that the algorithm can organize the data in c groups or clusters making use of its underlying structure. We use the algorithm to solve a binary classification problem. Therefore, we could set the number of desired clusters to two. However, normal traffic can be distributed in more than one cluster. We apply the algorithm several times, varying the number of considered clusters, and then define each cluster as normal or anomalous looking at the proportion of normal and attack records that it contains. In our problem, K-Means is an appropriate method for building up a classifier due to its unsupervised nature. However, as we are trying to detect anomalies and we should not know how anomalies are before we want to detect them, K-Means in its standard way can only be applied for a reactive model and not for a predictive one. But we can use K-Means in another way as well, instead of try to make different clusters with the whole set of data and then try to identify which clusters are for each class we can only give the algorithm the normal class data that we want to model. This produces different clusters for only what we know it is normal traffic. Once we have these clusters, we measure the distances from the normal records to the centroids of the method and then the distances from the attacks to the same centroids. Ideally, we should obtain larger distances for the attacks than the distances for the normal traces, as we have built the algorithm to minimize the distances to the normal traces. This is what we called in this paper the K-Means distance method. It is a pure anomaly detection algorithm because it is unsupervised and only needs one class to model the normal scenario.

Anomaly detection is very similar to novelty detection, which detects a sample that is different to an initial set of data. Considering novelty detection algorithms, there is a well known method that is a variation of the **Support Vector Machine** algorithm with the objective of obtaining a membership decision boundary for only one class of data. As SVMs are max-margin methods, this algorithm does not model a probability distribution of the data. It only finds a function that is positive for regions with high density of points and negative for small densities. In this work, both SVM approaches are included, so the classifier has been called SVM-2 and the novelty detector has been named as SVM-1.

A **Decision Tree** is a flowchart-like structure in which each internal node represents a "test" or a decision on an attribute. Each branch represents the outcome of the test, and each leaf node represents a class label. The full paths from root to leaf represent classification rules. We can train a decision tree structure with input train data and a desired class or label output in a supervised learning framework in order to adapt it to our problem. This simple algorithm is very convenient to compare with in this paper because we propose a new voting scheme algorithm which solution is in fact very similar to a decision tree structure.

A **Multilayer Perceptron** (MLP) is a type of feedforward artificial neural network. It is typically composed by three different layers. The first layer is called the input layer and it is fed with the numerical values that we want to be the input of the network. The second layer is called the hidden layer and it usually contains several nodes or neurons. Each neuron is fully connected to all of the input values of the first layer and it applies a non-linear activation function to a linear weighted combination of the inputs. The last layer is the output layer and it has the same number of nodes as values we want to estimate. When the network is used in a classification problem the output layer is supposed to give an approximated probability for every class we want to distinguish. MLP utilizes a supervised learning technique called backpropagation for training so it is a supervised algorithm. Moreover, the nonlinear activation functions of the layers make it able to distinguish data that is not linearly separable. Unlike the Decision tree algorithm this is a parametric method, and once optimized it offers a complex mathematical function that approximates the solution to the problem.

4 Experimental Setup

We conduct a set of experiments using the same original dataset described in the proposal. We perform three different transformation techniques to the data in order to cover all the possibilities. We contemplate all the combinations so at the end we obtain eight different datasets that are just numerical transformed versions of the original one. Figure 2 shows the full decision diagram of data transformations applied to the original dataset to obtain each of the used input datasets. The normalization, principal component parameters and the Feature Gaussian Mixture Probability Model latent components for each feature are computed only with the information of the normal records in the training set. Once adjusted, the three techniques are applied to our training and testing dataset without changing any configuration.

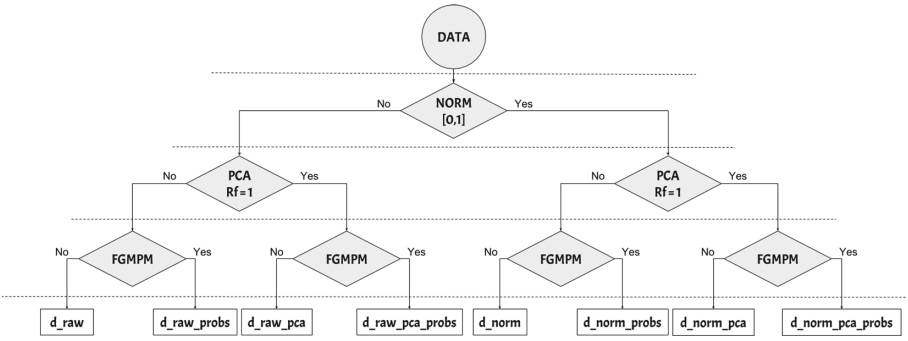

Fig. 2. Data transformation diagram

The eight generated datasets are:

- **d_raw**: The original NSL dataset without any transformation of the numerical values.
- **d_raw_probs**: We apply the FGMPM to the original NSL dataset values and change each feature value for the occurrence probability of each feature in the normal model.
- **d_raw_pca**: The uncorrelated version of the original NSL dataset with the same number of features.
- **d_raw_pca_probs**:We apply the FGMPM to the uncorrelated version of the original dataset and obtain the occurrence probabilities for this uncorrelated values of the features.
- **d_norm**: The original NSL dataset with the normal training values normalized to the range [0–1] and the remaining values normalized according to the previous scaler.
- **d_norm_probs**: We apply the FGPM to the normalized version of the dataset.
- **d_norm_pca**: The uncorrelated version of the normalized dataset.
- **d_norm_pca_probs**: The occurrence probabilities of the uncorrelated features of the normalized dataset.

For every mentioned dataset we build up six different models with the following algorithms:

- **Voting**: Our proposed voting scheme method for anomaly detection that can only be applied to the probability datasets.
- **KM-D**: The well known K-Means algorithm using the anomaly detection approach with the squared euclidean distances.
- **SVM**: A one class SVM for novelty detection.
- **KM-C**: K-Means algorithm in its standard clustering approach.
- **DT**: A default decision tree classifier.
- **MLP**: A simple multilayer perceptron with a hidden layer of 100 neurons and an output layer with 2 cells: attack or non-attack.

The first three algorithms are trained using only the normal training data due its anomaly detection objective and one class modeling capability. The KM-C is trained in an unsupervised way but using the normal and attack records of the training dataset. The DT and MLP are trained using the same data as the KM-C but in a supervised manner with the labels given. All models are tested with the whole NSL test dataset.

The experimental implementation has been developed using Python3.5 with the following libraries: Scikit-learn [18] version 0.20.2, Numpy version 1.13.0, Scipy version 0.19.0 and Pandas version 0.20.2. The FGMPM algorithm has been developed by the authors using pure Python mixed scikit-learn, which provides more flexibility in our research at the expense of less computational performance.

5 Results

We first introduce the set of metrics used to evaluate the performance of the proposed models with state of the art algorithms. Although we are facing an anomaly detection problem, our test can be considered as simple binary classification experiments: normal and attack traffic vectors. Therefore, we use the four basic metrics of the binary confusion matrix: True Positives (TP) and True Negatives (TN) for correctly classified records, and False Negatives (FN) and False Positives (FP) for the misclassified samples. These four values lead us to more interesting metrics in anomaly detection:

- *Sensitivity*: Positive detection rate.
- *Positive Predictive Value (PPV)*: True positives vs predicted positives rate.
- *Negative Predictive Value (NPV)*: True negatives vs predicted negatives rate.
- *F1 Score (F1)*: Harmonic mean between PPV and Sensitivity.
- *B*: Attack percentage in the whole testing dataset.
- *Intrusion Detection Capacity (CAP)*: A more complex and sensitive metric that relates the PPV and NPV with B and gives a very accurate idea of the complete performance of the model.

We select the three most interesting metrics for the anomaly detection systems for the evaluation: Sensitivity, in order to compare the anomaly detection rate, the F1-Score, as a measure of the test's accuracy, and Intrusion detection capacity (CAP), which best reflects the effectiveness of the models.

Table 1 shows the values obtained on the three selected metrics with every algorithm on every generated dataset. The detection methods are sorted, beginning with those who require less information to be trained. We highlight in red the experiments in which the algorithm does not converge to a valid solution. Also we have placed '-' where our proposed voting scheme makes no sense because the dataset is not composed by probabilities.

The SVM and the K-Means algorithms, in its classical approach, generally do not converge using not normalized data. It is an expected result, as both of them rely on the distances among the data. The K-Means algorithm, in its anomaly detection variant, does not converge well for the probabilities obtained after the PCA transformation. In general, supervised learning algorithms perform worse

Table 1. CAP, F1-Score and Sensitivity. E1 stands for d_norm, E2 for d_norm_probs, E3 for d_norm_pca, E4 for d_norm_pca_probs, E5 for d_raw, E6 for d_raw_probs, E7 for d_raw_pca and finally E8 for d_raw_pca_probs. The best results are highlighted in bold, whereas the experiments in which convergence was not achieved are in italic

		E1	**E2**	E3	**E4**	E5	**E6**	E7	**E8**
CAP	**Voting**	-	0,4714	-	**0,4972**	-	0,4797	-	**0,4958**
	KM-D	**0,4502**	0,3897	**0,4502**	*0,0306*	0,2155	0,4236	0,2155	*0,0405*
	SVM-1	0,3536	0,2011	0,3536	0,3536	*0,0022*	0,3067	*0,0023*	0,171
	KM-C	0,42	**0,5127**	0,4215	0,4097	*0,0005*	**0,5113**	*0,0005*	0,4772
	DT	0,3801	0,3456	0,3396	0,3347	**0,3934**	0,3659	**0,408**	0,3305
	SVM-2	0,3144	0,3557	0,3144	0,3144	0,3271	0,3366	0,2873	0,3216
	MLP	0,3505	0,305	0,3572	0,3318	0,3401	0,3027	0,3136	0,3333
F1	**Voting**	-	0,8703	-	**0,8838**	-	0,8715	-	**0,9061**
	KM-D	0,8558	0,8499	0,8558	*0,0163*	0,7169	0,8475	0,7169	*0,4066*
	SVM-1	0,7729	0,6255	0,7729	0,7729	*0,7372*	0,7303	*0,7373*	0,6177
	KM-C	**0,8631**	**0,8976**	**0,8636**	0,8826	*0,7369*	**0,8981**	*0,7369*	0,9054
	DT	0,7766	0,7504	0,7429	0,7458	**0,7877**	0,7681	**0,8015**	0,7364
	SVM-2	0,7171	0,7565	0,7171	0,7171	0,7568	0,7382	0,72	0,7678
	MLP	0,7513	0,7088	0,7575	0,8151	0,7796	0,7025	0,7972	0,7967
Sensitivity	**Voting**	-	0,7952	-	0,8184	-	0,7944	-	0,9032
	KM-D	0,7692	0,7865	0,7692	*0,0088*	0,5946	0,7616	0,5946	*0,2715*
	SVM-1	0,6397	0,4651	0,6397	0,6397	*0,9934*	0,5839	*0,9938*	0,4635
	KM-C	**0,8046**	**0,8512**	**0,8048**	**0,8913**	*0,9999*	**0,8539**	*0,9999*	**0,9495**
	DT	0,6392	0,6055	0,5954	0,6008	0,6545	0,6287	0,6746	0,5875
	SVM-2	0,5628	0,6128	0,5628	0,5628	0,6203	0,589	0,5733	0,5836
	MLP	0,6059	0,553	0,6139	0,7323	**0,6555**	0,5447	**0,7018**	0,6911

than unsupervised algorithms in our experiment. This is because of the fact that the testing dataset has different attacks than the training dataset, so the supervised algorithms cannot generalize as good as unsupervised ones. Although supervised algorithms seems to be always worse, they have a very high specificity, higher than unsupervised algorithms. The best result in the table is achieved by the K-Means clusters for the d_raw_pca_probs dataset. However, our voting scheme has a higher CAP than KM-C with this data because the overall performance of the model is better, although the KM-C has a higher anomaly detection rate. KM-C and our Voting scheme are the best algorithms here, but we have to consider that KM-C needs attacks in its training phase and does not provide a strictly normal model. On the other hand, our voting scheme offers a good normal model but it always needs the occurrence probabilities to be computed. Regarding the two hyperparameters of the Voting scheme, the best performance is achieved for a value of alpha equal to 0,013 and a consensus of 5. It has been noticed that an alpha increase can bee compensated with a consensus decrease and vice-versa in order to achieve a good performance. The KM-D algorithm seems to be a good alternative when we cannot normalize the data nor compute these probabilites.

6 Conclusions

Considering anomaly detection, unsupervised models are better suited to the real scenario, with unknown or untagged attacks or anomalies in datasets. We evaluate the impact of different preprocessing on the anomaly detection performance of different algorithms. We consider normalization, PCA and the probabilities of normal features with GMM. Moreover, we propose a Voting scheme algorithm. We train and test our voting scheme and multiple known unsupervised algorithms. The best results are obtained using KM-C and our Voting scheme, although the latter requires less information than the former. Considering the preprocessing, normalized data usually leads to a better performance. However, using the probabilities of normal features with GMM in NIDS with NSL-KDD, not normalized data generates more accurate probabilites and more sensitive detection algorithms. The PCA slightly improves the sensitivity of the anomaly detection algorithms, while it seems to have less effect with supervised algorithms. Finally, we have proved that using the occurrence probabilities improves the performance of the anomaly detection models and, specially, it allows the usage of a simple voting scheme to achieve a very good detector with F1-scores over 0.88 and CAP over 0.49, better than other more complex algorithms evaluated.

Acknowledgements. This work was supported by the Spanish Ministry of Economy and Competitiveness under contracts TIN-2015-65277-R, AYA2015-65973-C3-3-R and RTC-2016-5434-8.

References

1. Axelsson, S.: Intrusion detection systems: a survey and taxonomy. Chalmers University of Technology, Tech. rep. (2000)
2. Bahrololum, M., Khaleghi, M.: Anomaly intrusion detection system using Gaussian mixture model. In: 2008 Third International Conference on Convergence and Hybrid Information Technology, November 2008, vol. 1, pp. 1162–1167. https://doi.org/10.1109/ICCIT.2008.17
3. Barkan, O., Averbuch, A.: Robust mixture models for anomaly detection. In: 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP), September 2016, pp. 1–6. https://doi.org/10.1109/MLSP.2016.7738885
4. Breunig, M.M., Kriegel, H., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. In: Chen, W., Naughton, J.F., Bernstein, P.A. (eds.) Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, 16–18 May 2000, Dallas, Texas, USA, pp. 93–104. ACM (2000). https://doi.org/10.1145/342009.335388
5. Denning, D.E.: An intrusion-detection model. IEEE Trans. Softw. Eng. **13**(2), 222–232 (1987). https://doi.org/10.1109/TSE.1987.232894
6. Domingues, R., Filippone, M., Michiardi, P., Zouaoui, J.: A comparative evaluation of outlier detection algorithms: experiments and analyses. Pattern Recogn. **74**, 406–421 (2018)

7. Dromard, J., Roudière, G., Owezarski, P.: Online and scalable unsupervised network anomaly detection method. IEEE Trans. Netw. Serv. Manage. **14**(1), 34–47 (2017). https://doi.org/10.1109/TNSM.2016.2627340
8. Heady, R., Luger, G., Maccabe, A., Servilla, M.: The architecture of a network level intrusion detection system. Tech. rep., Los Alamos National Lab., NM, United States, New Mexico University, Albuquerque (1990)
9. Hock, D., Kappes, M.: A self-learning network anomaly detection system using majority voting. In: Dowland, P., Furnell, S., Ghita, B.V. (eds.) Proceedings Tenth International Network Conference, INC 2014, Plymouth, UK, 8–10 July 2014, pp. 59–69. Plymouth University (2014). http://www.cscan.org/openaccess/?paperid=225
10. Hodge, V.J., Austin, J.: A survey of outlier detection methodologies. Artif. Intell. Rev. **22**(2), 85–126 (2004). https://doi.org/10.1007/s10462-004-4304-y
11. Kdd cup 1999, October 2007. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
12. Kim, J., Scott, C.D.: Robust kernel density estimation. J. Mach. Learn. Res. **13**(1), 2529–2565 (2012). http://dl.acm.org/citation.cfm?id=2503308.2503323
13. Kukielka, P., Kotulski, Z.: Analysis of neural networks usage for detection of a new attack in IDS. Ann. UMCS Inf. **10**(1), 51–59 (2010)
14. Liu, D., Lung, C., Lambadaris, I., Seddigh, N.: Network traffic anomaly detection using clustering techniques and performance comparison. In: 2013 26th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), May 2013, pp. 1–4. https://doi.org/10.1109/CCECE.2013.6567739
15. Moustafa, N., Slay, J.: UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: Military Communications and Information Systems Conference (MilCIS), pp. 1–6. IEEE Stream (2015)
16. Moustafa, N., Slay, J.: The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Inf. Secur. J. A Global Perspect. **25**(1–13), 1–14 (2016)
17. NSL-KDD data set for network-based intrusion detection systems, March 2009. http://nsl.cs.unb.ca/NSL-KDD/
18. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
19. Revathi, S., Malathi, A.: A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. Int. J. Eng. Res. Tech. **2**(12), 1848–1853 (2013)
20. Reynolds, D.D.: Gaussian Mixture Models. In: Li, S.Z., Jain, A. (eds.) Encyclopedia of Biometrics. Springer, Boston (2009). https://doi.org/10.1007/978-0-387-73003-5
21. Shahreza, M.L., Moazzami, D., Moshiri, B., Delavar, M.: Anomaly detection using a self-organizing map and particle swarm optimization. Scientia Iranica **18**(6), 1460–1468 (2011). https://doi.org/10.1016/j.scient.2011.08.025
22. Zhang, R., Zhang, S., Muthuraman, S., Jiang, J.: One class support vector machine for anomaly detection in the communication network performance data. In: Proceedings of the 5th Conference on Applied Electromagnetics, Wireless and Optical Communications, pp. 31–37. ELECTROSCIENCE'07, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point (2007)